НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп'ютерних систем**

**Лабораторна робота №2**

з дисципліни **Бази даних і засоби управління**

*на тему: "Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL"*

Виконала:
студент III курсу
групи КВ-94
Холодар А. А.
Перевірив:
Петрашенко А. В.

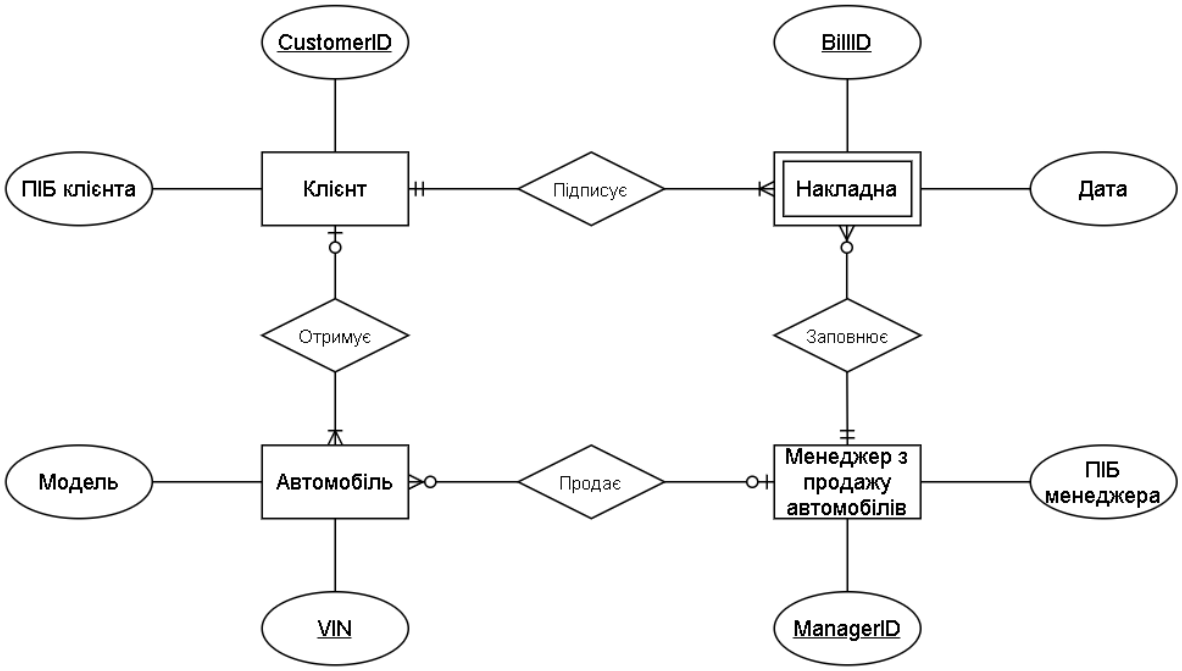
Київ – 2021

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

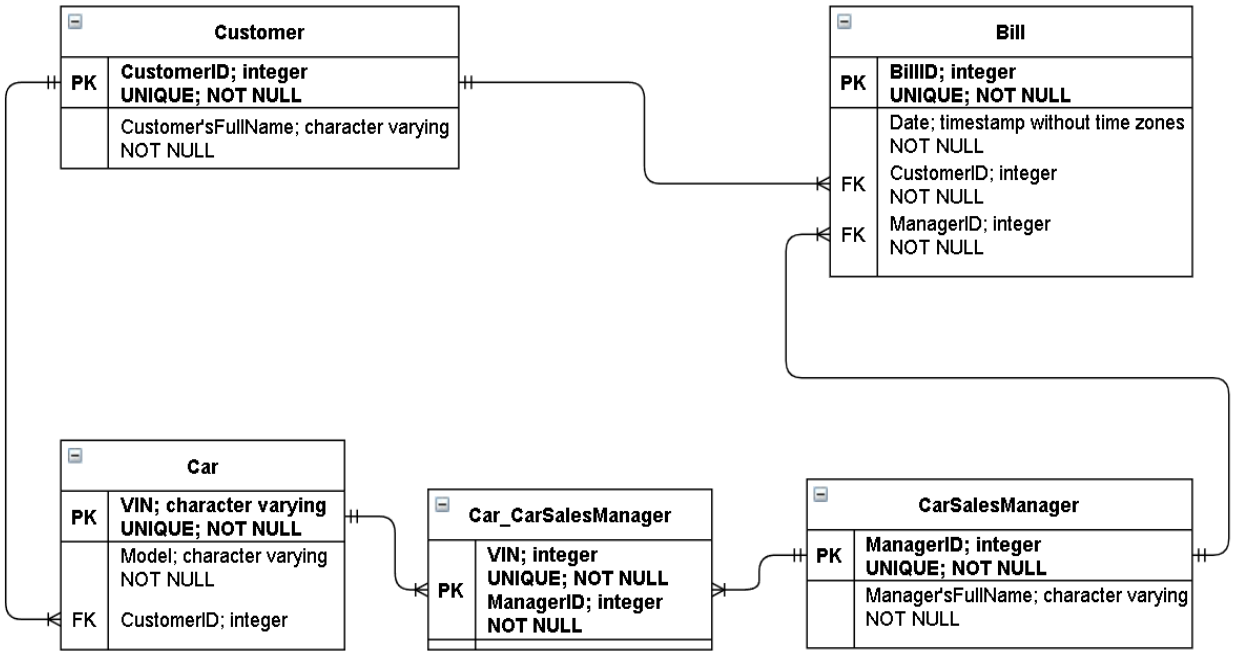
*Загальне завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

## *Модель «сутність-зв'язок» галузі продажу автомобілей автосалоном*




## *Перетворення моделі у схему бази даних*



### *Опис бази даних:*

Обрана предметна галузь передбачає зберігання та продаж автомобілів. Згідно цієї області для побудови бази даних було виділено наступні сутності:

1. «Клієнт», з атрибутами: «ПІБ клієнта», «CustomerID». Призначена для того, щоб фіксувати людину, яка здійснила купівлю автомобіля в автосалоні.
2. «Накладна», з атрибутами: «дата», «BillID». Призначена для збереження дати оформлення продажу автомобіля.
3. «Менеджер з продажу автомобілів», з атрибутами: «ПІБ менеджера», «ManagerID». Призначена для фіксування того, який менеджер займався продажем автомобіля.
4. «Автомобіль», з атрибутами: «Модель», «VIN». Призначена для збереження інформації, яка стосується автомобілів у наявності та проданих.

### *Опис меню програми:*

```
1 -> Show one table
2 -> Show all tables
3 -> Insert information
4 -> Delete information
5 -> Update information
6 -> Random information
7 -> Select information
8 -> Exit
```

1. Show one table – вивід на екран однієї з таблиць
2. Show all tables – вивід на екран усіх таблиць БД.
3. Insert information – вставка нового рядка у певну таблицю.
4. Delete information – видалення рядка з певної таблиці.
5. Update information – оновлення даних рядка у певній таблиці.
6. Random information – запис рандомізованих даних у таблицю.
7. Select information – запити для фільтрації таблиць трьома способами.
8. Exit – завершення роботи програми.

## *Завдання 1*

*Insert*

(на прикладі відношення “Customer” -> “Bill”)

Початковий вигляд таблиць:

```
---------------------------------------
-------------Customer----------------
---------------------------------------
CustomerID = 3
Customer'sFullName = YuppuLover
---------------------------------------
CustomerID = 1
Customer'sFullName = Proger
---------------------------------------
CustomerID = 2
Customer'sFullName = sdfzxcmbv
---------------------------------------
CustomerID = 4
Customer'sFullName = zxcgenius
---------------------------------------
```

```
---------------------------------------
---------------Bill-----------------
---------------------------------------
BillID = 1
Date = 2019-07-26 00:00:00
CustomerID = 1
ManagerID = 1
---------------------------------------
BillID = 2
Date = 2017-06-06 00:00:00
CustomerID = 2
ManagerID = 1
---------------------------------------
BillID = 3
Date = 2016-05-05 00:00:00
CustomerID = 1
ManagerID = 1
---------------------------------------
```

Запис до таблиці “Customer”:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 1
Input CustomerID: 5
Input Customer'sFullName: Genius Prostous
['ЗАМЕЧАНИЕ:  inserted\n']
```

```
---------------------------------------
-------------Customer----------------
---------------------------------------
CustomerID = 3
Customer'sFullName = YuppuLover
---------------------------------------
CustomerID = 1
Customer'sFullName = Proger
---------------------------------------
CustomerID = 2
Customer'sFullName = sdfzxcmbv
---------------------------------------
CustomerID = 4
Customer'sFullName = zxcgenius
---------------------------------------
CustomerID = 5
Customer'sFullName = Genius Prostous
---------------------------------------
```

Спроба запису до таблиці рядка з первинним ключем, який вже там знаходиться:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 1
Input CustomerID: 5
Input Customer'sFullName: Just User
['ЗАМЕЧАНИЕ:  "CustomerID" = 5 is existing already.\n']
```

Запис до таблиці “Bill”:

```
                                    ------------------------------------
                                    ----------------Bill----------------
                                    ------------------------------------
                                    BillID = 1
                                    Date = 2019-07-26 00:00:00
                                    CustomerID = 1
        1 -> Customer               ManagerID = 1
        2 -> Bill                   ------------------------------------
        3 -> Car                    BillID = 2
        4 -> CarSalesManager        Date = 2017-06-06 00:00:00
        5 -> Car_CarSalesManager    CustomerID = 2
                                    ManagerID = 1
                                    ------------------------------------
                                    BillID = 3
Please, input table number: 2       Date = 2016-05-05 00:00:00
Input BillID: 4                     CustomerID = 1
Input Date: 2020-05-21              ManagerID = 1
Input CustomerID: 1                 ------------------------------------
Input ManagerID: 1                  BillID = 4
['ЗАМЕЧАНИЕ:  inserted\n']          Date = 2020-05-21 00:00:00
                                    CustomerID = 1
                                    ManagerID = 1
                                    ------------------------------------
```

Спроба запису до таблиці рядка з вторинним ключем, який не відповідає первинному таблиці “Customer” та спроба запису до таблиці рядка з первинним ключем, який вже існує:

```
Please, input table number: 2
Input BillID: 5
Input Date: 2020-09-09
Input CustomerID: 29
Input ManagerID: 1
['ЗАМЕЧАНИЕ:  "CustomerID" = 29 is not exist or "BillID" = 5 is existing already or "ManagerID" = 1 is not exist.\n']

                        1 -> Continue insertion in this table
                        2 -> Stop insertion in this table

Your choice -> 1
Input BillID: 4
Input Date: 2020-09-09
Input CustomerID: 1
Input ManagerID: 1
['ЗАМЕЧАНИЕ:  "CustomerID" = 1 is not exist or "BillID" = 4 is existing already or "ManagerID" = 1 is not exist.\n']
```

## ***Insert-лістинг:***

```
@staticmethod
def insertToCustomer(custid, fullname, goodnews):
    badnews = '"CustomerID" = {} is existing already.'.format(custid)
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if not exists (select "CustomerID" from "Customer"
where "CustomerID" = {}) ' \
             'then insert into "Customer"("CustomerID", "Customer\'\'sFullName")
VALUES ({},\'{}\'); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$;'.format(custid, custid, fullname, goodnews,
badnews)
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def insertToBill(custid, billid, manid, date, goodnews):
    badnews = '"CustomerID" = {} is not exist or "BillID" = {} is existing
already or "ManagerID" = {} is not exist.'.format(custid, billid, manid)
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if exists (select "CustomerID" from "Customer" where
"CustomerID" = {}) and not exists ' \
             '(select "BillID" from "Bill" where "BillID" = {}) and exists
(select "ManagerID" from "CarSalesManager" where "ManagerID" = {}) then ' \
             'insert into "Bill"("BillID", "Date", "CustomerID", "ManagerID")
values ({}, \'{}\', {}, {}); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(custid, billid, manid, billid, date,
custid, manid, goodnews, badnews)
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def insertToCar(vin, mod, custid, goodnews):
    badnews = '"VIN" = "{}" is existing or "CustomerID" = {} is not
existing.'.format(vin, custid)
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if exists (select "CustomerID" from "Customer" where
"CustomerID" = {}) ' \
             'and not exists (select "VIN" from "Car" where "VIN" = \'{}\') then
' \
             'insert into "Car"("VIN", "Model", "CustomerID") values (\'{}\',
\'{}\', {}); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(custid, vin, vin, mod, custid, goodnews,
badnews)
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
```

```
def insertToCarSalesManager(manid, manfullname, goodnews):
    badnews = '"ManagerID" = {} is existing already.'.format(manid)
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if not exists (select "ManagerID" from
"CarSalesManager" where "ManagerID" = {}) ' \
             'then insert into "CarSalesManager"("ManagerID",
"Manager\'\'sFullName") values ({}, \'{}\'); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(manid, manid, manfullname, goodnews,
badnews)
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def insertToCar_CarSalesManager(saleid, manid, vin, goodnews):
    badnews = '"ManagerID" = {} is not exist or "SaleID" = {} is existing
already or "VIN" = "{}" is not exist.'.format(manid, saleid, vin)
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if exists (select "ManagerID" from "CarSalesManager"
where "ManagerID" = {}) and ' \
             'exists (select "VIN" from "Car" where "VIN" = \'{}\') and not
exists (select "SaleID" from "Car_CarSalesManager" where "SaleID" = {}) ' \
             'then insert into "Car_CarSalesManager"("SaleID", "ManagerID",
"VIN") values ({}, {}, \'{}\'); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(manid, vin, saleid, saleid, manid, vin,
goodnews, badnews)
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)
```

## *Delete*

(на прикладі відношення “Car” -> “Car_CarSalesManager”)

Початковий вигляд таблиць:

```
--------------------------------------
----------------Car-------------------
--------------------------------------
VIN = GSDAFHNKLLZXCJ45456JHDFGJ6DJ165DF56
Model = A
CustomerID = 1
--------------------------------------
VIN = PSDFADFSJHDFZXCHA45654645SGFHSHFD65SDFH5456SDH
Model = Skoda
CustomerID = 1
--------------------------------------
VIN = PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH
Model = Kobra Striker
CustomerID = 1
--------------------------------------
VIN = AFHSDGFJHLFGSJDKSJDGFHLJLK456654
Model = AFKORDS
CustomerID = 1
--------------------------------------
```

```
--------------------------------------
--------Car_CarSalesManager-----------
--------------------------------------
SaleID = 1
ManagerID = 1
VIN = PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH
--------------------------------------
SaleID = 2
ManagerID = 1
VIN = GSDAFHNKLLZXCJ45456JHDFGJ6DJ165DF56
--------------------------------------
```

Видалення рядка, у якого нема залежності первинного ключа у інших таблицях:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 3
Input VIN: AFHSDGFJHLFGSJDKSJDGFHLJLK456654
['ЗАМЕЧАНИЕ:  deleted\n']
```

```
--------------------------------------
----------------Car-------------------
--------------------------------------
VIN = GSDAFHNKLLZXCJ45456JHDFGJ6DJ165DF56
Model = A
CustomerID = 1
--------------------------------------
VIN = PSDFADFSJHDFZXCHA45654645SGFHSHFD65SDFH5456SDH
Model = Skoda
CustomerID = 1
--------------------------------------
VIN = PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH
Model = Kobra Striker
CustomerID = 1
--------------------------------------
```

Спроба видалення рядка, у якого є залежності або якого не існує:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 3
Input VIN: PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH
['ЗАМЕЧАНИЕ:  "VIN" = "PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH" is not exist or there are some dependencies on it.\n']

                        1 -> Continue delete in this table
                        2 -> Stop delete in this table

Your choice -> 1
Input VIN: PROMUA
['ЗАМЕЧАНИЕ:  "VIN" = "PROMUA" is not exist or there are some dependencies on it.\n']
```

## ***Delete-лістинг:***

```
@staticmethod
def deleteInCustomer(custid, goodnews):
    badnews = '"CustomerID" = {} is not exist or there are some dependencies on
it.'.format(custid)
    connect = controller.connection()
    cursor = connect.cursor()
    delete = 'do $$ begin if exists (select "CustomerID" from "Customer" where
"CustomerID" = {}) ' \
             'and not exists (select "CustomerID" from "Bill" where "CustomerID"
= {}) and not exists ' \
             '(select "CustomerID" from "Car" where "CustomerID" = {}) then ' \
             'delete from "Customer" where "CustomerID" = {}; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(custid, custid, custid, custid, goodnews,
badnews)
    cursor.execute(delete)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def deleteInBill(billid, goodnews):
    badnews = '"BillID" = {} is not exist.'.format(billid)
    connect = controller.connection()
    cursor = connect.cursor()
    delete = 'do $$ begin if exists (select "BillID" from "Bill" where "BillID"
= {}) ' \
             'then delete from "Bill" where "BillID" = {}; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(billid, billid, goodnews, badnews)
    cursor.execute(delete)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)


@staticmethod
def deleteInCar(vin, goodnews):
    badnews = '"VIN" = "{}" is not exist or there are some dependencies on
it.'.format(vin)
    connect = controller.connection()
    cursor = connect.cursor()
    delete = 'do $$ begin if exists (select "VIN" from "Car" where "VIN" =
\'{}\') and ' \
             'not exists (select "VIN" from "Car_CarSalesManager" where "VIN" =
\'{}\') then ' \
             'delete from "Car" where "VIN" = \'{}\'; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '. format(vin, vin, vin, goodnews, badnews)
    cursor.execute(delete)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def deleteInCarSalesManager(manid, goodnews):
    badnews = '"ManagerID" = {} is not exist or there are some dependencies on
it.'.format(manid)
```

```
    connect = controller.connection()
    cursor = connect.cursor()
    delete = 'do $$ begin if exists (select "ManagerID" from "CarSalesManager"
where "ManagerID" = {}) and ' \
             'not exists (select "ManagerID" from "Bill" where "ManagerID" = {})
and not exists ' \
             '(select "ManagerID" from "Car_CarSalesManager" where "ManagerID" =
{}) then ' \
             'delete from "CarSalesManager" where "ManagerID" = {}; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '. format(manid, manid, manid, manid, goodnews,
badnews)
    cursor.execute(delete)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def deleteInCar_CarSalesManager(saleid, goodnews):
    badnews = '"SaleID" = {} is not exist.'.format(saleid)
    connect = controller.connection()
    cursor = connect.cursor()
    delete = 'do $$ begin if exists (select "SaleID" from "Car_CarSalesManager"
where "SaleID" = {}) then ' \
             'delete from "Car_CarSalesManager" where "SaleID" = {};' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(saleid, saleid, goodnews, badnews)
    cursor.execute(delete)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)
```

*Update*

(на прикладі таблиці “CarSalesManager”)

Початковий вигляд таблиці:

```
--------------------------------------
----------CarSalesManager-------------
--------------------------------------
ManagerID = 1
Manager'sFullName = ProGamer
--------------------------------------
```

Спроба редагування:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 4
Input ManagerID: 1
Input Manager'sFullName: Poroshenko Petro
['ЗАМЕЧАНИЕ:  updated\n']
```

```
--------------------------------------
----------CarSalesManager-------------
--------------------------------------
ManagerID = 1
Manager'sFullName = Poroshenko Petro
--------------------------------------
```

Спроба редагування рядка, якого не існує:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 4
Input ManagerID: 2
Input Manager'sFullName: Proger3000
['ЗАМЕЧАНИЕ:  "ManagerID" = "2" is not exist\n']
```

## ***Update-лістинг:***

```
@staticmethod
def updateCustomer(custid, custfullname, goodnews):
    badnews = '"CustomerID" = {} is not exist.'.format(custid)
    connect = controller.connection()
    cursor = connect.cursor()
    update = 'do $$ begin if exists (select "CustomerID" from "Customer" where
"CustomerID" = {}) then ' \
             'update "Customer" set "Customer\'\'sFullName" = \'{}\' where
"CustomerID" = {}; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(custid, custfullname, custid, goodnews,
badnews)
    cursor.execute(update)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def updateBill(billid, date, goodnews):
    badnews = '"BillID" = {} is not exist'.format(billid)
    connect = controller.connection()
    cursor = connect.cursor()
    update = 'do $$ begin if exists (select "BillID" from "Bill" where "BillID"
= {}) then ' \
             'update "Bill" set "Date" = \'{}\' where "BillID" = {}; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(billid, date, billid, goodnews, badnews)
    cursor.execute(update)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def updateCar(vin, mod, goodnews):
    badnews = '"VIN" = "{}" is not exist'.format(vin)
    connect = controller.connection()
    cursor = connect.cursor()
    update = 'do $$ begin if exists (select "VIN" from "Car" where "VIN" =
\'{}\') then ' \
             'update "Car" set "Model" = \'{}\' where "VIN" = \'{}\'; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(vin, mod, vin, goodnews, badnews)
    cursor.execute(update)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def updateCarSalesManager(manid, manfullname, goodnews):
    badnews = '"ManagerID" = "{}" is not exist'.format(manid)
    connect = controller.connection()
    cursor = connect.cursor()
    update = 'do $$ begin if exists (select "ManagerID" from "CarSalesManager"
where "ManagerID" = {}) then ' \
             'update "CarSalesManager" set "Manager\'\'sFullName" = \'{}\' where
"ManagerID" = {}; ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
```

```
            'end if; end $$; '.format(manid, manfullname, manid, goodnews,
badnews)
    cursor.execute(update)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)
```

***Завдання 2***

(на прикладі таблиці “CarSalesManager”)

Передбачити автоматичне пакетне генерування “рандомізованих” даних:

Початковий вигляд таблиці:

```
--------------------------------------
----------CarSalesManager-------------
--------------------------------------
ManagerID = 1
Manager'sFullName = Poroshenko Petro
--------------------------------------
```

Спроба генерування рандомізованих даних:

```
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager

Please, input table number: 4
Input count of new elements: 3
SQL request -> do $$ begin if (1=1)
['ЗАМЕЧАНИЕ:  inserted randomly\n']
```

```
--------------------------------------
----------CarSalesManager-------------
--------------------------------------
ManagerID = 1
Manager'sFullName = Poroshenko Petro
--------------------------------------
ManagerID = 79710
Manager'sFullName = OPASDFGHJK
--------------------------------------
ManagerID = 53453
Manager'sFullName = TYUIOPASDF
--------------------------------------
ManagerID = 48205
Manager'sFullName = xcvbnmQWER
--------------------------------------
```

## *Лістинг:*

```
@staticmethod
def randomToCustomer(num, goodnews, badnews):
    connect = controller.connection()
    cursor = connect.cursor()
    insert =  'do $$ begin if (1=1) then ' \
              'insert into "Customer"("CustomerID", "Customer\'\'sFullName")
select random()*99999, ' \
              'substr(characters, (random() * length(characters) + 1)::integer,
10) ' \
              'from
(VALUES(\'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM\')) as
symbols(characters), generate_series(1, {}); ' \
              'raise notice \'{}\'; ' \
              'else raise notice \'{}\'; ' \
              'end if; end $$; '.format(num, goodnews, badnews)
    controller.msg("SQL request -> {}".format(insert))
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def randomToBill(num, goodnews, badnews):
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if  (1=1) then ' \
              'insert into "Bill"("BillID", "Date", "CustomerID", "ManagerID")
select random()*99999, ' \
              'timestamp \'2020-01-10\' - random() * (timestamp \'2020-01-20\' -
timestamp \'2021-01-10\'), ' \
              '(select "CustomerID" from "Customer" order by random() limit 1), '
\
              '(select "ManagerID" from "CarSalesManager" order by random() limit
1) ' \
              'from generate_series(1, {}); ' \
              'raise notice \'{}\'; ' \
              'else raise notice \'{}\'; ' \
              'end if; end $$; '.format(num, goodnews, badnews)
    controller.msg("SQL request -> {}".format(insert))
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def randomToCar(num, goodnews, badnews):
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if (1=1) then ' \
              'insert into "Car"("VIN", "Model", "CustomerID") select
substr(characters, (random() * length(characters) + 1)::integer, 10), ' \
              'substr(characters, (random() * length(characters) + 1)::integer,
6), (select "CustomerID" from "Customer" order by random() limit 1) ' \
              'from
(VALUES(\'qwertyuiopasdfghjklzxcvbnm123456789QWERTYUIOPASDFGHJKLZXCVBNM\')) as
symbols(characters), generate_series(1, {}); ' \
              'raise notice \'{}\'; ' \
              'else raise notice \'{}\'; ' \
              'end if; end $$; '.format(num, goodnews, badnews)
    controller.msg("SQL request -> {}".format(insert))
    cursor.execute(insert)
    connect.commit()
```

```
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)

@staticmethod
def randomToCarSalesManager(num, goodnews, badnews):
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if (1=1) then ' \
             'insert into "CarSalesManager"("ManagerID", "Manager\'\'sFullName")
select random()*99999, ' \
             'substr(characters, (random() * length(characters) + 1)::integer,
10) ' \
             'from
(VALUES(\'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM\')) as
symbols(characters), generate_series(1, {}); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(num, goodnews, badnews)
    controller.msg("SQL request -> {}".format(insert))
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)


@staticmethod
def randomToCar_CarSalesManager(num, goodnews, badnews):
    connect = controller.connection()
    cursor = connect.cursor()
    insert = 'do $$ begin if (1=1) then ' \
             'insert into "Car_CarSalesManager"("SaleID", "ManagerID", "VIN")
select random()*99999, ' \
             '(select "ManagerID" from "CarSalesManager" order by random() limit
1), ' \
             '(select "VIN" from "Car" order by random() limit 1) from
generate_series(1, {}); ' \
             'raise notice \'{}\'; ' \
             'else raise notice \'{}\'; ' \
             'end if; end $$; '.format(num, goodnews, badnews)
    controller.msg("SQL request -> {}".format(insert))
    cursor.execute(insert)
    connect.commit()
    controller.msg(connect.notices)
    cursor.close()
    controller.disconnection(connect)
```

## *Завдання 3*

Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно.

```
---------------------------------------
1 -> Show Customer'sFullName, VIN and Model where VIN includes some letter and where length of model name more or equal then some number
---------------------------------------
---------------------------------------
2 -> Show the history of certain manager
---------------------------------------
---------------------------------------
3 -> Show the history of certain customer
---------------------------------------
```

Дані для першого виклику:

Таблиця “Customer”:

| | CustomerID [PK] integer | Customer'sFullName character varying |
|---|---|---|
| 1 | 1 | Proger |
| 2 | 2 | sdfzxcmbv |
| 3 | 3 | YuppuLover |
| 4 | 4 | zxcgenius |
| | | |

Таблиця “Car”:

| | VIN [PK] character varying | Model character varying | CustomerID integer |
|---|---|---|---|
| 1 | AFHSDGFJHLFGSJDKSJDGFHLJLK456654 | AFKORDS | 1 |
| 2 | GSDAFHNKLLZXCJ45456JHDFGJ6DJ165DF56 | A | 1 |
| 3 | PSDFADFSJHDFZXCHA45654645SGFHSHFD65SDFH5456SDH | Skoda | 1 |
| 4 | PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH | Kobra Striker | 1 |
| | | | |

Виконання запиту:

| | Customer'sFullName character varying | VIN character varying | Model character varying |
|---|---|---|---|
| 1 | Proger | PSDFADFSJHDFZXCHA45654645SGFHSHFD65SDFH5456SDH | Skoda |
| 2 | Proger | PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH | Kobra Striker |

```
Your choice -> 1
Input letters: ZXC
Enter number: 2
Time of request 1 ms
[]
---------------------------------------
----------------foo--------------------
---------------------------------------
Customer'sFullName = Proger
VIN = PSDFADFSJHDFZXCHA45654645SGFHSHFD65SDFH5456SDH
Model = Skoda
---------------------------------------
Customer'sFullName = Proger
VIN = PSDFAGASHAHSZXC45SGFHSHFD65SDFH5456SDH
Model = Kobra Striker
```

Дані для другого виклику:

Таблиця “Bill”:

| | BillID [PK] integer | Date timestamp without time zone | CustomerID integer | ManagerID integer |
|---|---|---|---|---|
| 1 | 1 | 2019-07-26 00:00:00 | 1 | 1 |
| 2 | 2 | 2017-06-06 00:00:00 | 2 | 1 |

Таблиця “CarSalesManager”:

| | ManagerID [PK] integer | Manager'sFullName character varying |
|---|---|---|
| 1 | 1 | ProGamer |

Виконання запиту:

| | Manager'sFullName character varying | BillID integer | Date timestamp without time zone |
|---|---|---|---|
| 1 | ProGamer | 1 | 2019-07-26 00:00:00 |
| 2 | ProGamer | 2 | 2017-06-06 00:00:00 |

```
Your choice -> 2
Input Manager'sFullName: ProGamer
Time of request 2 ms
[]
----------------------------------------
----------------foo---------------------
----------------------------------------
Manager'sFullName = ProGamer
BillID = 1
Date = 2019-07-26 00:00:00
----------------------------------------
Manager'sFullName = ProGamer
BillID = 2
Date = 2017-06-06 00:00:00
----------------------------------------
----------------------------------------
```

Дані для третього виклику:

Таблиця “Bill”:

| | BillID [PK] integer | Date timestamp without time zone | CustomerID integer | ManagerID integer |
|---|---|---|---|---|
| 1 | 1 | 2019-07-26 00:00:00 | 1 | 1 |
| 2 | 2 | 2017-06-06 00:00:00 | 2 | 1 |
| 3 | 3 | 2016-05-05 00:00:00 | 1 | 1 |

Таблиця “Customer”:

| | CustomerID [PK] integer | Customer'sFullName character varying |
|---|---|---|
| 1 | 1 | Proger |
| 2 | 2 | sdfzxcmbv |
| 3 | 3 | YuppuLover |
| 4 | 4 | zxcgenius |
| | | |

Виконання запиту:

| | Customer'sFullName character varying | BillID integer | Date timestamp without time zone |
|---|---|---|---|
| 1 | Proger | 1 | 2019-07-26 00:00:00 |
| 2 | Proger | 3 | 2016-05-05 00:00:00 |

```
Your choice -> 3
Input Customer'sFullName: Proger
Time of request 2 ms
[]
--------------------------------------
----------------foo-------------------
--------------------------------------
Customer'sFullName = Proger
BillID = 1
Date = 2019-07-26 00:00:00
--------------------------------------
Customer'sFullName = Proger
BillID = 3
Date = 2016-05-05 00:00:00
--------------------------------------
```

## *Код програми*

### model.py

```
import controller
import time

tables = {
    1: 'Customer',
    2: 'Bill',
    3: 'Car',
    4: 'CarSalesManager',
    5: 'Car_CarSalesManager',
}

class Model:

    @staticmethod
    def show_one_table(table):
        connect = controller.connection()
        cursor = connect.cursor()
        show = 'select * from public."{}"'.format(tables[table])
        print("SQL request => ", show)
        print('')
        cursor.execute(show)
        records = cursor.fetchall()
        cursor.close()
        controller.disconnection(connect)
        return records


    @staticmethod
    def insertToCustomer(custid, fullname, goodnews):
        badnews = '"CustomerID" = {} is existing already.'.format(custid)
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if not exists (select "CustomerID" from "Customer"
where "CustomerID" = {}) ' \
                 'then insert into "Customer"("CustomerID",
"Customer\'\'sFullName") VALUES ({},\'{}\'); ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$;'.format(custid, custid, fullname, goodnews,
badnews)
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def insertToBill(custid, billid, manid, date, goodnews):
        badnews = '"CustomerID" = {} is not exist or "BillID" = {} is existing
already or "ManagerID" = {} is not exist.'.format(custid, billid, manid)
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if exists (select "CustomerID" from "Customer"
where "CustomerID" = {}) and not exists ' \
                 '(select "BillID" from "Bill" where "BillID" = {}) and exists
(select "ManagerID" from "CarSalesManager" where "ManagerID" = {}) then ' \
                 'insert into "Bill"("BillID", "Date", "CustomerID",
"ManagerID") values ({}, \'{}\', {}, {}); ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(custid, billid, manid, billid, date,
custid, manid, goodnews, badnews)
        cursor.execute(insert)
```

```
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def insertToCar(vin, mod, custid, goodnews):
        badnews = '"VIN" = "{}" is existing or "CustomerID" = {} is not
existing.'.format(vin, custid)
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if exists (select "CustomerID" from "Customer"
where "CustomerID" = {}) ' \
                 'and not exists (select "VIN" from "Car" where "VIN" = \'{}\')
then ' \
                 'insert into "Car"("VIN", "Model", "CustomerID") values
(\'{}\', \'{}\', {}); ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(custid, vin, vin, mod, custid,
goodnews, badnews)
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def insertToCarSalesManager(manid, manfullname, goodnews):
        badnews = '"ManagerID" = {} is existing already.'.format(manid)
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if not exists (select "ManagerID" from
"CarSalesManager" where "ManagerID" = {}) ' \
                 'then insert into "CarSalesManager"("ManagerID",
"Manager\'\'sFullName") values ({}, \'{}\'); ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(manid, manid, manfullname, goodnews,
badnews)
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def insertToCar_CarSalesManager(saleid, manid, vin, goodnews):
        badnews = '"ManagerID" = {} is not exist or "SaleID" = {} is existing
already or "VIN" = "{}" is not exist.'.format(manid, saleid, vin)
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if exists (select "ManagerID" from
"CarSalesManager" where "ManagerID" = {}) and ' \
                 'exists (select "VIN" from "Car" where "VIN" = \'{}\') and not
exists (select "SaleID" from "Car_CarSalesManager" where "SaleID" = {}) ' \
                 'then insert into "Car_CarSalesManager"("SaleID", "ManagerID",
"VIN") values ({}, {}, \'{}\'); ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(manid, vin, saleid, saleid, manid,
vin, goodnews, badnews)
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
```

```
        controller.disconnection(connect)


    @staticmethod
    def deleteInCustomer(custid, goodnews):
        badnews = '"CustomerID" = {} is not exist or there are some dependencies
on it.'.format(custid)
        connect = controller.connection()
        cursor = connect.cursor()
        delete = 'do $$ begin if exists (select "CustomerID" from "Customer"
where "CustomerID" = {}) ' \
                 'and not exists (select "CustomerID" from "Bill" where
"CustomerID" = {}) and not exists ' \
                 '(select "CustomerID" from "Car" where "CustomerID" = {}) then
' \
                 'delete from "Customer" where "CustomerID" = {}; ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(custid, custid, custid, custid,
goodnews, badnews)
        cursor.execute(delete)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def deleteInBill(billid, goodnews):
        badnews = '"BillID" = {} is not exist.'.format(billid)
        connect = controller.connection()
        cursor = connect.cursor()
        delete = 'do $$ begin if exists (select "BillID" from "Bill" where
"BillID" = {}) ' \
                 'then delete from "Bill" where "BillID" = {}; ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(billid, billid, goodnews, badnews)
        cursor.execute(delete)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)


    @staticmethod
    def deleteInCar(vin, goodnews):
        badnews = '"VIN" = "{}" is not exist or there are some dependencies on
it.'.format(vin)
        connect = controller.connection()
        cursor = connect.cursor()
        delete = 'do $$ begin if exists (select "VIN" from "Car" where "VIN" =
\'{}\') and ' \
                 'not exists (select "VIN" from "Car_CarSalesManager" where
"VIN" = \'{}\') then ' \
                 'delete from "Car" where "VIN" = \'{}\'; ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '. format(vin, vin, vin, goodnews, badnews)
        cursor.execute(delete)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def deleteInCarSalesManager(manid, goodnews):
```

```
        badnews = '"ManagerID" = {} is not exist or there are some dependencies
on it.'.format(manid)
        connect = controller.connection()
        cursor = connect.cursor()
        delete = 'do $$ begin if exists (select "ManagerID" from
"CarSalesManager" where "ManagerID" = {}) and ' \
                 'not exists (select "ManagerID" from "Bill" where "ManagerID" =
{}) and not exists ' \
                 '(select "ManagerID" from "Car_CarSalesManager" where
"ManagerID" = {}) then ' \
                 'delete from "CarSalesManager" where "ManagerID" = {}; ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '. format(manid, manid, manid, manid,
goodnews, badnews)
        cursor.execute(delete)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def deleteInCar_CarSalesManager(saleid, goodnews):
        badnews = '"SaleID" = {} is not exist.'.format(saleid)
        connect = controller.connection()
        cursor = connect.cursor()
        delete = 'do $$ begin if exists (select "SaleID" from
"Car_CarSalesManager" where "SaleID" = {}) then ' \
                 'delete from "Car_CarSalesManager" where "SaleID" = {};' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(saleid, saleid, goodnews, badnews)
        cursor.execute(delete)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def updateCustomer(custid, custfullname, goodnews):
        badnews = '"CustomerID" = {} is not exist.'.format(custid)
        connect = controller.connection()
        cursor = connect.cursor()
        update = 'do $$ begin if exists (select "CustomerID" from "Customer"
where "CustomerID" = {}) then ' \
                 'update "Customer" set "Customer\'\'sFullName" = \'{}\' where
"CustomerID" = {}; ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(custid, custfullname, custid,
goodnews, badnews)
        cursor.execute(update)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def updateBill(billid, date, goodnews):
        badnews = '"BillID" = {} is not exist'.format(billid)
        connect = controller.connection()
        cursor = connect.cursor()
        update = 'do $$ begin if exists (select "BillID" from "Bill" where
"BillID" = {}) then ' \
                 'update "Bill" set "Date" = \'{}\' where "BillID" = {}; ' \
                 'raise notice \'{}\'; ' \
```

```
                    'else raise notice \'{}\'; ' \
                    'end if; end $$; '.format(billid, date, billid, goodnews,
badnews)
            cursor.execute(update)
            connect.commit()
            controller.msg(connect.notices)
            cursor.close()
            controller.disconnection(connect)

    @staticmethod
    def updateCar(vin, mod, goodnews):
            badnews = '"VIN" = "{}" is not exist'.format(vin)
            connect = controller.connection()
            cursor = connect.cursor()
            update = 'do $$ begin if exists (select "VIN" from "Car" where "VIN" =
\'{}\') then ' \
                    'update "Car" set "Model" = \'{}\' where "VIN" = \'{}\'; ' \
                    'raise notice \'{}\'; ' \
                    'else raise notice \'{}\'; ' \
                    'end if; end $$; '.format(vin, mod, vin, goodnews, badnews)
            cursor.execute(update)
            connect.commit()
            controller.msg(connect.notices)
            cursor.close()
            controller.disconnection(connect)

    @staticmethod
    def updateCarSalesManager(manid, manfullname, goodnews):
            badnews = '"ManagerID" = "{}" is not exist'.format(manid)
            connect = controller.connection()
            cursor = connect.cursor()
            update = 'do $$ begin if exists (select "ManagerID" from
"CarSalesManager" where "ManagerID" = {}) then ' \
                    'update "CarSalesManager" set "Manager\'\'sFullName" = \'{}\'
where "ManagerID" = {}; ' \
                    'raise notice \'{}\'; ' \
                    'else raise notice \'{}\'; ' \
                    'end if; end $$; '.format(manid, manfullname, manid, goodnews,
badnews)
            cursor.execute(update)
            connect.commit()
            controller.msg(connect.notices)
            cursor.close()
            controller.disconnection(connect)

    @staticmethod
    def select_first(vin, modlen):
            connect = controller.connection()
            cursor = connect.cursor()
            select = 'select "Customer\'\'sFullName", "VIN", "Model" from (select
c."Customer\'\'sFullName", p."VIN", p."Model" ' \
                    'from "Customer" c inner join "Car" p on p."CustomerID" =
c."CustomerID" where p."VIN" like \'%{}%\' and length(p."Model") >= {} ' \
                    'group by c."Customer\'\'sFullName", p."VIN", p."Model") as
foo'.format(vin, modlen)
            beg = int(time.time() * 1000)
            cursor.execute(select)
            end = int(time.time() * 1000) - beg
            records = cursor.fetchall()
            print('Time of request {} ms'.format(end))
            controller.msg(connect.notices)
            cursor.close()
            controller.disconnection(connect)
            return records

    @staticmethod
```

```
    def select_second(manfullname):
        connect = controller.connection()
        cursor = connect.cursor()
        select = 'select "Manager\'\'sFullName", "BillID", "Date" from (select
c."Manager\'\'sFullName", p."BillID", p."Date" ' \
                 'from "CarSalesManager" c inner join "Bill" p on p."ManagerID"
= c."ManagerID" where c."Manager\'\'sFullName" like ' \
                 '\'{}\' group by c."Manager\'\'sFullName", p."BillID",
p."Date") as foo'. format(manfullname)
        beg = int(time.time() * 1000)
        cursor.execute(select)
        end = int(time.time() * 1000) - beg
        records = cursor.fetchall()
        print('Time of request {} ms'.format(end))
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)
        return records

    @staticmethod
    def select_third(custfullname):
        connect = controller.connection()
        cursor = connect.cursor()
        select = 'select "Customer\'\'sFullName", "BillID", "Date" from (select
c."Customer\'\'sFullName", p."BillID", p."Date" ' \
                 'from "Customer" c inner join "Bill" p on p."CustomerID" =
c."CustomerID" where c."Customer\'\'sFullName" like ' \
                 '\'{}\' group by c."Customer\'\'sFullName", p."BillID",
p."Date") as foo'.format(custfullname)
        beg = int(time.time() * 1000)
        cursor.execute(select)
        end = int(time.time() * 1000) - beg
        records = cursor.fetchall()
        print('Time of request {} ms'.format(end))
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)
        return records

    @staticmethod
    def randomToCustomer(num, goodnews, badnews):
        connect = controller.connection()
        cursor = connect.cursor()
        insert =  'do $$ begin if (1=1) then ' \
                  'insert into "Customer"("CustomerID", "Customer\'\'sFullName")
select random()*99999, ' \
                 'substr(characters, (random() * length(characters) +
1)::integer, 10) ' \
                 'from
(VALUES(\'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM\')) as
symbols(characters), generate_series(1, {}); ' \
                 'raise notice \'{}\'; ' \
                  'else raise notice \'{}\'; ' \
                  'end if; end $$; '.format(num, goodnews, badnews)
        controller.msg("SQL request -> {}".format(insert))
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def randomToBill(num, goodnews, badnews):
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if  (1=1) then ' \
```

```
                'insert into "Bill"("BillID", "Date", "CustomerID",
"ManagerID") select random()*99999, ' \
                'timestamp \'2020-01-10\' - random() * (timestamp \'2020-01-
20\' - timestamp \'2021-01-10\'), ' \
                '(select "CustomerID" from "Customer" order by random() limit
1), ' \
                '(select "ManagerID" from "CarSalesManager" order by random()
limit 1) ' \
                'from generate_series(1, {}); ' \
                'raise notice \'{}\'; ' \
                'else raise notice \'{}\'; ' \
                'end if; end $$; '.format(num, goodnews, badnews)
        controller.msg("SQL request -> {}".format(insert))
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def randomToCar(num, goodnews, badnews):
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if (1=1) then ' \
                'insert into "Car"("VIN", "Model", "CustomerID") select
substr(characters, (random() * length(characters) + 1)::integer, 10), ' \
                'substr(characters, (random() * length(characters) +
1)::integer, 6), (select "CustomerID" from "Customer" order by random() limit 1)
' \
                'from
(VALUES(\'qwertyuiopasdfghjklzxcvbnm123456789QWERTYUIOPASDFGHJKLZXCVBNM\')) as
symbols(characters), generate_series(1, {}); ' \
                'raise notice \'{}\'; ' \
                'else raise notice \'{}\'; ' \
                'end if; end $$; '.format(num, goodnews, badnews)
        controller.msg("SQL request -> {}".format(insert))
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)

    @staticmethod
    def randomToCarSalesManager(num, goodnews, badnews):
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if (1=1) then ' \
                'insert into "CarSalesManager"("ManagerID",
"Manager\'\'sFullName") select random()*99999, ' \
                'substr(characters, (random() * length(characters) +
1)::integer, 10) ' \
                'from
(VALUES(\'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM\')) as
symbols(characters), generate_series(1, {}); ' \
                'raise notice \'{}\'; ' \
                'else raise notice \'{}\'; ' \
                'end if; end $$; '.format(num, goodnews, badnews)
        controller.msg("SQL request -> {}".format(insert))
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)


    @staticmethod
```

```
    def randomToCar_CarSalesManager(num, goodnews, badnews):
        connect = controller.connection()
        cursor = connect.cursor()
        insert = 'do $$ begin if (1=1) then ' \
                 'insert into "Car_CarSalesManager"("SaleID", "ManagerID",
"VIN") select random()*99999, ' \
                 '(select "ManagerID" from "CarSalesManager" order by random()
limit 1), ' \
                 '(select "VIN" from "Car" order by random() limit 1) from
generate_series(1, {}); ' \
                 'raise notice \'{}\'; ' \
                 'else raise notice \'{}\'; ' \
                 'end if; end $$; '.format(num, goodnews, badnews)
        controller.msg("SQL request -> {}".format(insert))
        cursor.execute(insert)
        connect.commit()
        controller.msg(connect.notices)
        cursor.close()
        controller.disconnection(connect)
```

## view.py

```
import controller
import pandas as pd
from model import Model

class View:
    def __init__(self, table, records):
        self.table = table
        self.records = records

    @staticmethod
    def input_item(item):
        data = input("Input {}: ".format(item))
        return data


    @staticmethod
    def item_entering(item):
        data = input('Enter {}: '.format(item))
        return data

    @staticmethod
    def list():
        print('''
        1 -> Customer
        2 -> Bill
        3 -> Car
        4 -> CarSalesManager
        5 -> Car_CarSalesManager
        ''')

    @staticmethod
    def listofcolumn(table):
        if table == 1:
            print('''
            1 -> Customer'sFullName
            ''')
        elif table == 2:
            print('''
            1 -> Date
            2 -> CustomerID
            3 -> ManagerID
            ''')
        elif table == 3:
            print('''
            1 -> Model
            2 -> CustomerID
            ''')
        elif table == 4:
            print('''
            1 -> Manager'sFullName
            ''')
        elif table == 5:
            print('''
            1 -> ManagerID
            2 -> VIN
            ''')

    def show_table(self):
        print('---------------------------------------')
        if self.table == 1:
            print('--------------Customer----------------')
            print('---------------------------------------')
            for row in self.records:
                print('CustomerID = {}'.format(row[0]))
```

```
            print('Customer\'sFullName = {}'.format(row[1]))
            print('----------------------------------------')
    elif self.table == 2:
        print('----------------Bill------------------')
        print('----------------------------------------')
        for row in self.records:
            print('BillID = {}'.format(row[0]))
            print('Date = {}'.format(row[1]))
            print('CustomerID = {}'.format(row[2]))
            print('ManagerID = {}'.format(row[3]))
            print('----------------------------------------')
    elif self.table == 3:
        print('----------------Car-------------------')
        print('----------------------------------------')
        for row in self.records:
            print('VIN = {}'.format(row[0]))
            print('Model = {}'.format(row[1]))
            print('CustomerID = {}'.format(row[2]))
            print('----------------------------------------')
    elif self.table == 4:
        print('----------CarSalesManager-------------')
        print('----------------------------------------')
        for row in self.records:
            print('ManagerID = {}'.format(row[0]))
            print('Manager\'sFullName = {}'.format(row[1]))
            print('----------------------------------------')
    elif self.table == 5:
        print('--------Car_CarSalesManager-----------')
        print('----------------------------------------')
        for row in self.records:
            print('SaleID = {}'.format(row[0]))
            print('ManagerID = {}'.format(row[1]))
            print('VIN = {}'.format(row[2]))
            print('----------------------------------------')


def select_show_table(self):
    print('----------------------------------------')
    if self.table == 1:
        print('----------------foo-------------------')
        print('----------------------------------------')
        for row in self.records:
            print('Customer\'sFullName = {}'.format(row[0]))
            print('VIN = {}'.format(row[1]))
            print('Model = {}'.format(row[2]))
            print('----------------------------------------')

    elif self.table == 2:
        print('----------------foo-------------------')
        print('----------------------------------------')
        for row in self.records:
            print('Manager\'sFullName = {}'.format(row[0]))
            print('BillID = {}'.format(row[1]))
            print('Date = {}'.format(row[2]))
            print('----------------------------------------')

    elif self.table == 3:
        print('----------------foo-------------------')
        print('----------------------------------------')
        for row in self.records:
            print('Customer\'sFullName = {}'.format(row[0]))
            print('BillID = {}'.format(row[1]))
            print('Date = {}'.format(row[2]))
            print('----------------------------------------')
```

```
class Menu:
    @staticmethod
    def menu():
        while True:
            print('''
            1 -> Show one table
            2 -> Show all tables
            3 -> Insert information
            4 -> Delete information
            5 -> Update information
            6 -> Random information
            7 -> Select information
            8 -> Exit
            ''')
            choice = input('Please, make a choice: ')
            if choice.isdigit():
                choice = int(choice)
                if choice == 1:
                    View.list()
                    numoftable = controller.valid_table()
                    records = Model.show_one_table(numoftable)
                    obj = View(numoftable, records)
                    obj.show_table()
                elif choice == 2:
                    for i in range(1, 6):
                        records = Model.show_one_table(i)
                        obj = View(i, records)
                        obj.show_table()
                elif choice == 3:
                    View.list()
                    numoftable = controller.valid_table()
                    goodnews = 'inserted'
                    cicle = True
                    while cicle:
                        if numoftable == 1:
                            custid = View.input_item("CustomerID")
                            custfullname =
View.input_item("Customer\'sFullName")
                            row = [custid, custfullname]
                            istrue = controller.valid_data_to_insert(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.insertToCustomer(custid, custfullname,
goodnews)
                        elif numoftable == 2:
                            billid = View.input_item("BillID")
                            date = View.input_item("Date")
                            custid = View.input_item("CustomerID")
                            manid = View.input_item("ManagerID")
                            row = [billid, date, custid, manid]
                            istrue = controller.valid_data_to_insert(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.insertToBill(custid, billid, manid, date,
goodnews)
                        elif numoftable == 3:
                            vin = View.input_item("VIN")
                            mod = View.input_item("Model")
                            custid = View.input_item("CustomerID")
```

```
                            row = [vin, mod, custid]
                            istrue = controller.valid_data_to_insert(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.insertToCar(vin, mod, custid, goodnews)
                        elif numoftable == 4:
                            manid = View.input_item("ManagerID")
                            manfullname = View.input_item("Manager\'sFullName")
                            row = [manid, manfullname]
                            istrue = controller.valid_data_to_insert(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.insertToCarSalesManager(manid,
manfullname, goodnews)
                        elif numoftable == 5:
                            saleid = View.input_item("SaleID")
                            manid = View.input_item("ManagerID")
                            vin = View.input_item("VIN")
                            row = [saleid, manid, vin]
                            istrue = controller.valid_data_to_insert(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.insertToCar_CarSalesManager(saleid, manid,
vin, goodnews)
                        cont = True
                        while cont:
                            print('''
                            1 -> Continue insertion in this table
                            2 -> Stop insertion in this table
                            ''')
                            ch = input('Your choice -> ')
                            if ch == '2':
                                cicle = False
                                cont = False
                            elif ch == '1':
                                cont = False
                                pass
                            else:
                                print('Try again')
                            continue
                    elif choice == 4:
                        View.list()
                        numoftable = controller.valid_table()
                        goodnews = 'deleted'
                        cicle = True
                        while cicle:
                            if numoftable == 1:
                                custid = View.input_item("CustomerID")
                                istrue = controller.valid_data_to_delete(custid)
                                if istrue == 0:
                                    controller.msg('Your data are invalid. Try
again.')
                                else:
                                    Model.deleteInCustomer(custid, goodnews)

                            elif numoftable == 2:
                                billid = View.input_item("BillID")
```

```
                            istrue = controller.valid_data_to_delete(billid)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.deleteInBill(billid, goodnews)

                        elif numoftable == 3:
                            vin = View.input_item("VIN")
                            Model.deleteInCar(vin, goodnews)

                        elif numoftable == 4:
                            manid = View.input_item("ManagerID")
                            istrue = controller.valid_data_to_delete(manid)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.deleteInCarSalesManager(manid, goodnews)

                        elif numoftable == 5:
                            saleid = View.input_item("SaleID")
                            istrue = controller.valid_data_to_delete(saleid)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.deleteInCar_CarSalesManager(saleid,
goodnews)
                        cont = True
                        while cont:
                            print('''
                            1 -> Continue delete in this table
                            2 -> Stop delete in this table
                            ''')
                            ch = input('Your choice -> ')
                            if ch == '2':
                                cicle = False
                                cont = False
                            elif ch == '1':
                                cont = False
                                pass
                            else:
                                print('Try again.')

                elif choice == 5:
                    View.list()
                    numoftable = controller.valid_table()
                    goodnews = 'updated'
                    cicle = True
                    while cicle:
                        if numoftable == 1:
                            custid = View.input_item("CustomerID")
                            custfullname =
View.input_item("Customer\'sFullName")
                            row = [custid, custfullname]
                            istrue = controller.valid_data_to_update(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.updateCustomer(custid, custfullname,
goodnews)
                        elif numoftable == 2:
                            billid = View.input_item("BillID")
```

```
                            date = View.input_item("Date")
                            row = [billid, date]
                            istrue = controller.valid_data_to_update(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.updateBill(billid, date, goodnews)

                        elif numoftable == 3:
                            vin = View.input_item("VIN")
                            mod = View.input_item("Model")
                            row = [vin, mod]
                            istrue = controller.valid_data_to_update(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.updateCar(vin, mod, goodnews)
                        elif numoftable == 4:
                            manid = View.input_item("ManagerID")
                            manfullname = View.input_item("Manager\'sFullName")
                            row = [manid, manfullname]
                            istrue = controller.valid_data_to_update(numoftable,
row)
                            if istrue == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.updateCarSalesManager(manid, manfullname,
goodnews)
                        elif numoftable == 5:
                            controller.msg('You can not update something in this
table. Sorry, try another.')
                            break
                        cont = True
                        while cont:
                            print('''
                            1 -> Continue update this table
                            2 -> Stop update this table
                            ''')
                            ch = input('Your choice -> ')
                            if ch == '2':
                                cicle = False
                                cont = False
                            elif ch == '1':
                                cont = False
                                pass
                            else:
                                print('Try again.')

                elif choice == 6:
                    View.list()
                    numoftable = controller.valid_table()
                    goodnews = 'inserted randomly'
                    badnews = 'something went wrong.'
                    cicle = True
                    while cicle:
                        if numoftable == 1:
                            num = View.input_item("count of new elements")
                            num = controller.valid_data_to_random(num)
                            if num == 0:
                                controller.msg('Your data are invalid. Try
again.')
```

```
                            else:
                                Model.randomToCustomer(num, goodnews, badnews)
                        elif numoftable == 2:
                            num = View.input_item("count of new elements")
                            num = controller.valid_data_to_random(num)
                            if num == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.randomToBill(num, goodnews, badnews)
                        elif numoftable == 3:
                            num = View.input_item("count of new elements")
                            num = controller.valid_data_to_random(num)
                            if num == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.randomToCar(num, goodnews, badnews)
                        elif numoftable == 4:
                            num = View.input_item("count of new elements")
                            num = controller.valid_data_to_random(num)
                            if num == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.randomToCarSalesManager(num, goodnews,
badnews)
                        elif numoftable == 5:
                            num = View.input_item("count of new elements")
                            num = controller.valid_data_to_random(num)
                            if num == 0:
                                controller.msg('Your data are invalid. Try
again.')
                            else:
                                Model.randomToCar_CarSalesManager(num, goodnews,
badnews)
                        cont = True
                        while cont:
                            print('''
                            1 -> Continue random in this table
                            2 -> Stop random in this table
                            ''')
                            ch = input('Your choice -> ')
                            if ch == '2':
                                cicle = False
                                cont = False
                            elif ch == '1':
                                cont = False
                                pass
                            else:
                                print('Try again.')
                    elif choice == 7:
                        cicle = True
                        while cicle:
                            print('----------------------------------------')
                            print('1 -> Show Customer\'sFullName, VIN and Model
where VIN includes some letter and where length of model name more or equal then
some number')
                            print('----------------------------------------')
                            print('----------------------------------------')
                            print('2 -> Show the history of certain manager')
                            print('----------------------------------------')
                            print('----------------------------------------')
                            print('3 -> Show the history of certain customer')
                            print('----------------------------------------')
                            choice = controller.valid_choice_to_select()
```

```
                        if choice == 1:
                            letters = View.input_item("letters")
                            number = controller.valid_select_first()
                            records = Model.select_first(letters, number)
                            obj = View(choice, records)
                            obj.select_show_table()
                        elif choice == 2:
                            manfullname = View.input_item("Manager\'sFullName")
                            records = Model.select_second(manfullname)
                            obj = View(choice, records)
                            obj.select_show_table()
                        elif choice == 3:
                            custfullname =
View.input_item("Customer\'sFullName")
                            records = Model.select_third(custfullname)
                            obj = View(choice, records)
                            obj.select_show_table()
                        cont = True
                        while cont:
                            print('''
                            1 -> Continue selection
                            2 -> Stop selection
                            ''')
                            ch = input('Your choice -> ')
                            if ch == '2':
                                cicle = False
                                cont = False
                            elif ch == '1':
                                cont = False
                                pass
                            else:
                                print('Try again.')

                elif choice == 8:
                    controller.msg('Good bye!')
                    break
                else:
                    controller.msg("Try something another.")
            else:
                print('Please, enter the digit.')
```

## controller.py

```
import psycopg2
import view

def connection():
    return psycopg2.connect(
        user="postgres",
        password="qwerty",
        host="localhost",
        port="5432",
        database="antonybase",
    )
def disconnection(connection):
    connection.commit()
    connection.close()

def msg(text):
    return print(text)

def valid_table():
    while True:
        table = input ('Please, input table number: ')
        if table.isdigit():
            table = int(table)
            if table > 0 and table < 6:
                break
            else:
                print('Please, enter number from 1 to 5.')
        else:
            print('Please, enter a digit.')
    return table

def valid_data_to_insert(table, row):
    if table == 1:
        if str(row[0]).isdigit():
            return 1
        else:
            return 0
    elif table == 2:
        if str(row[0]).isdigit() and (len(row[1]) >= 10) and (row[1][4] == '-'
and row[1][7] == '-' and int(row[1][5]) <= 1 and int(row[1][8]) <= 3)\
                and str(row[2]).isdigit() and str(row[3]).isdigit():
            if int(row[1][5]) == 0 and  int(row[1][6]) == 1:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 2:
                if int(row[1][8]) >= 2 and  int(row[1][9]) > 8:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 3:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 4:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 5:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
```

```
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 6:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 7:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 8:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and  int(row[1][6]) == 9:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 1 and  int(row[1][6]) == 0:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 1 and  int(row[1][6]) == 1:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 1 and  int(row[1][6]) == 2:
                if int(row[1][8]) >= 3 and  int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            else:
                return 0

        else:
            return 0
    elif table == 3:
        if str(row[2]).isdigit():
            return 1
        else:
            return 0
    elif table == 4:
        if str(row[0]).isdigit():
            return 1
        else:
            return 0
    elif table == 5:
        if str(row[0]).isdigit() and str(row[1]).isdigit():
            return 1
        else:
            return 0


def valid_data_to_delete(id):
    if str(id).isdigit():
        return 1
    else:
        return 0
```

```
def valid_data_to_update(table, row):
    if table == 1:
        if str(row[0]).isdigit():
            return 1
        else:
            return 0
    elif table == 2:
        if str(row[0]).isdigit() and (len(row[1]) >= 10) and (
                row[1][4] == '-' and row[1][7] == '-' and int(row[1][5]) <= 1
and int(row[1][8]) <= 3):
            if int(row[1][5]) == 0 and int(row[1][6]) == 1:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 2:
                if int(row[1][8]) >= 2 and int(row[1][9]) > 8:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 3:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 4:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 5:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 6:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 7:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 8:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 0 and int(row[1][6]) == 9:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 1 and int(row[1][6]) == 0:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 1 and int(row[1][6]) == 1:
                if int(row[1][8]) >= 3 and int(row[1][9]) > 0:
                    return 0
                else:
                    return 1
            elif int(row[1][5]) == 1 and int(row[1][6]) == 2:
```

```
                if int(row[1][8]) >= 3 and int(row[1][9]) > 1:
                    return 0
                else:
                    return 1
            else:
                return 0

        else:
            return 0
    elif table == 3:
            return 1
    elif table == 4:
        if str(row[0]).isdigit():
            return 1
        else:
            return 0



def valid_choice_to_select():
    while True:
        choice = input('Your choice -> ')
        if choice.isdigit():
            choice = int(choice)
            if choice > 0 and choice < 4:
                break
            else:
                print('Please, enter number from 1 to 3.')
        else:
            print('Please, enter a digit.')
    return choice

def valid_select_first():
    while True:
        number = input('Enter number: ')
        if number.isdigit():
            number = int(number)
            return number
        else:
            print('Please, enter a digit.')

def valid_data_to_random(id):
    if str(id).isdigit():
        id = int(id)
        return id
    else:
        return 0
```

## main.py

```
import psycopg2
import controller
from view import Menu

try:
    Menu.menu()
except (Exception , psycopg2.Error) as error :
        controller.msg("PostgreSQL Error: {}".format(error))
finally:
    controller.msg("PostgreSQL connection is closed")
```